100

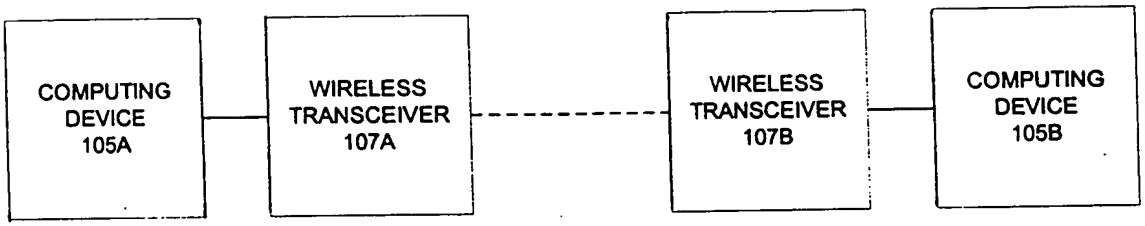
COMPUTING DEVICE 105A
WIRELESS TRANSCEIVER 107A
WIRELESS TRANSCEIVER 107B
COMPUTING DEVICE 105B

Figure 1

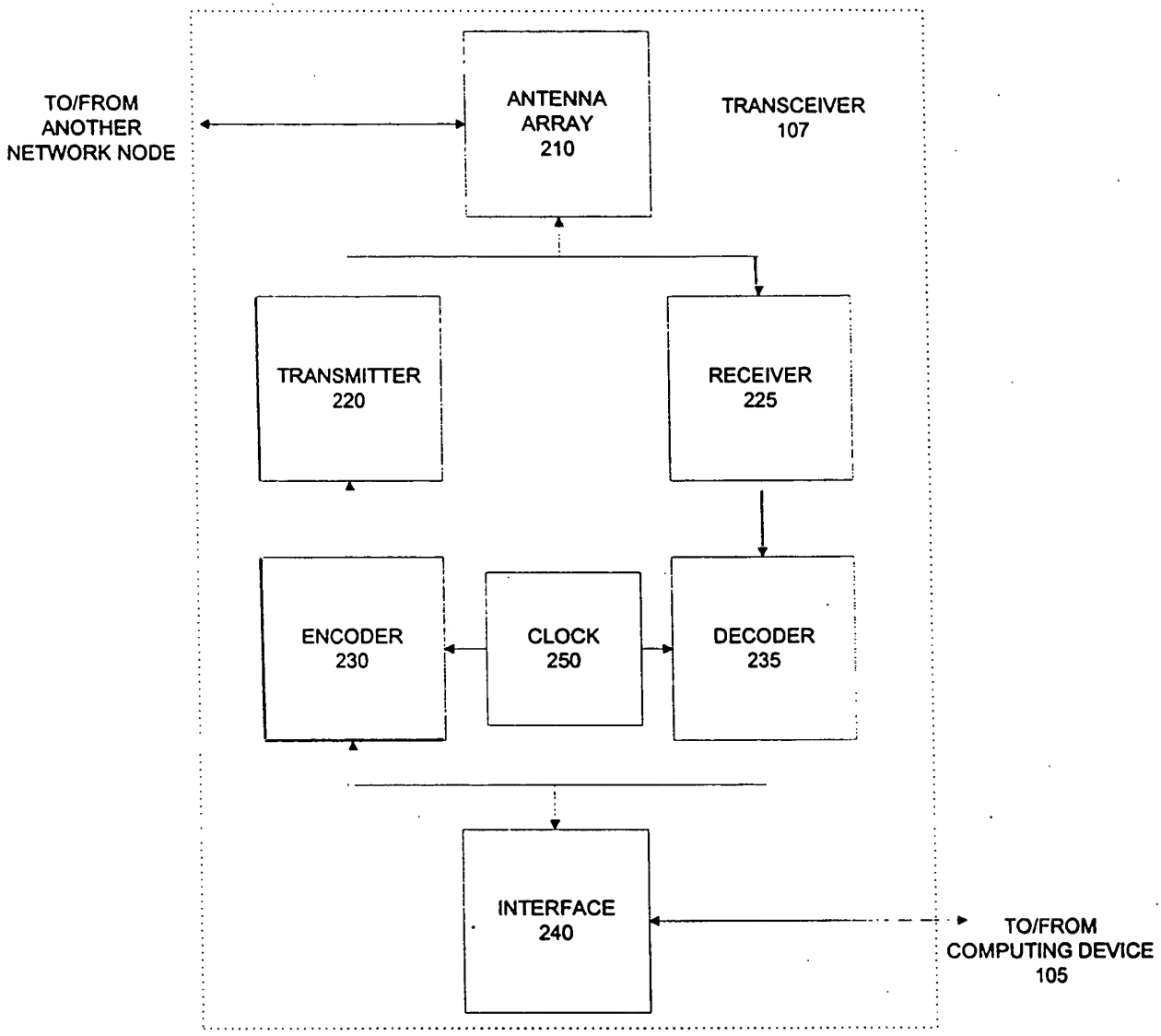
TO/FROM ANOTHER NETWORK NODE
ANTENNA ARRAY 210
TRANSCEIVER 107
TRANSMITTER 220
RECEIVER 225
ENCODER 230
CLOCK 250
DECODER 235
INTERFACE 240
TO/FROM COMPUTING DEVICE 105

Figure 2

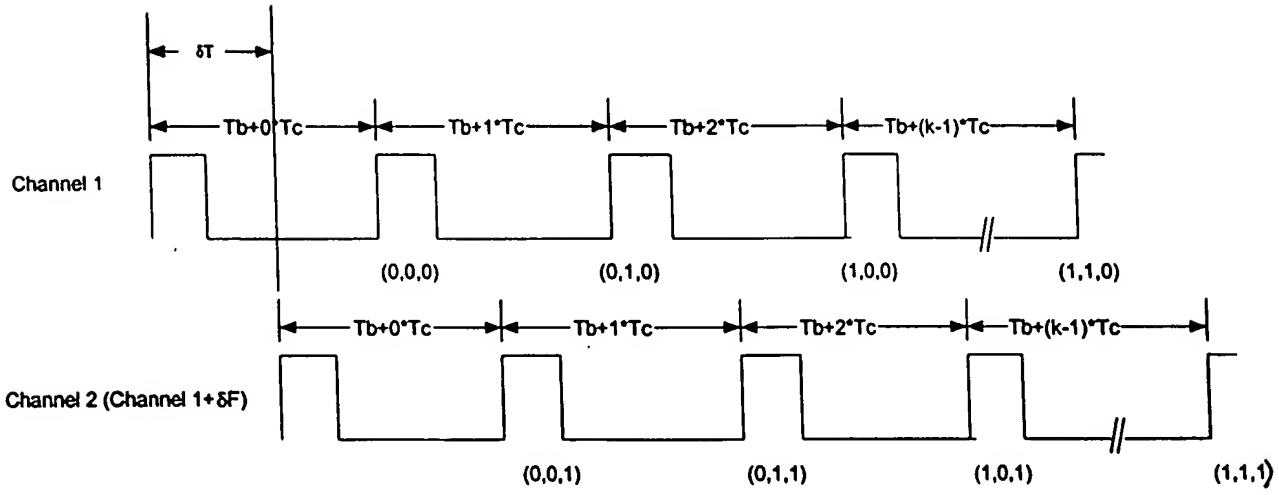

Figure 3

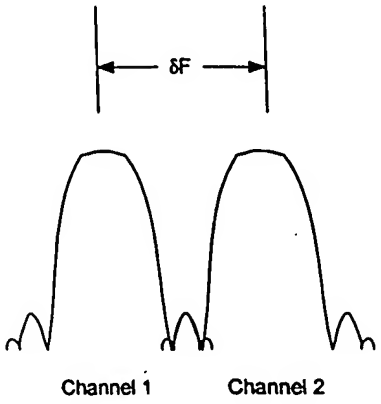

Figure 4

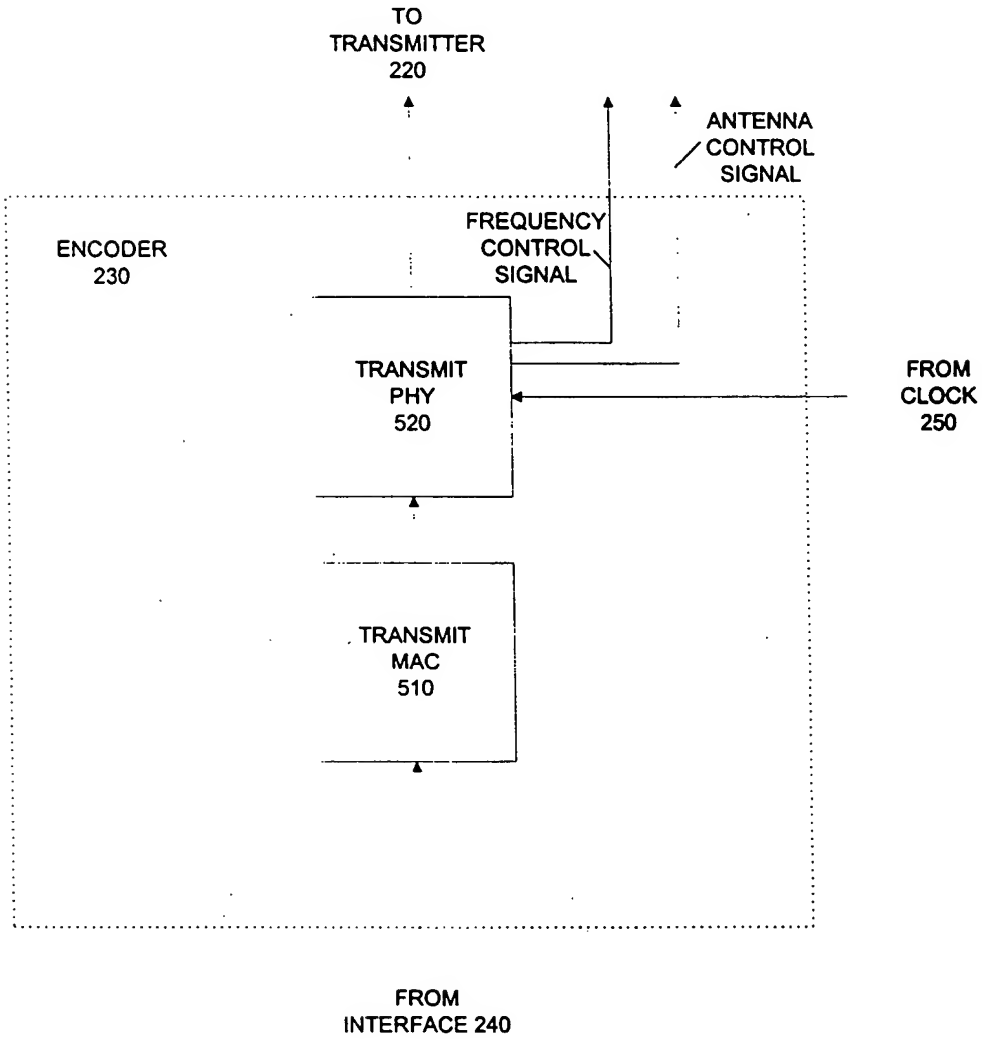
TO TRANSMITTER 220
ANTENNA CONTROL SIGNAL
FREQUENCY CONTROL SIGNAL
ENCODER 230
TRANSMIT PHY 520
FROM CLOCK 250
TRANSMIT MAC 510
FROM INTERFACE 240

Figure 5A

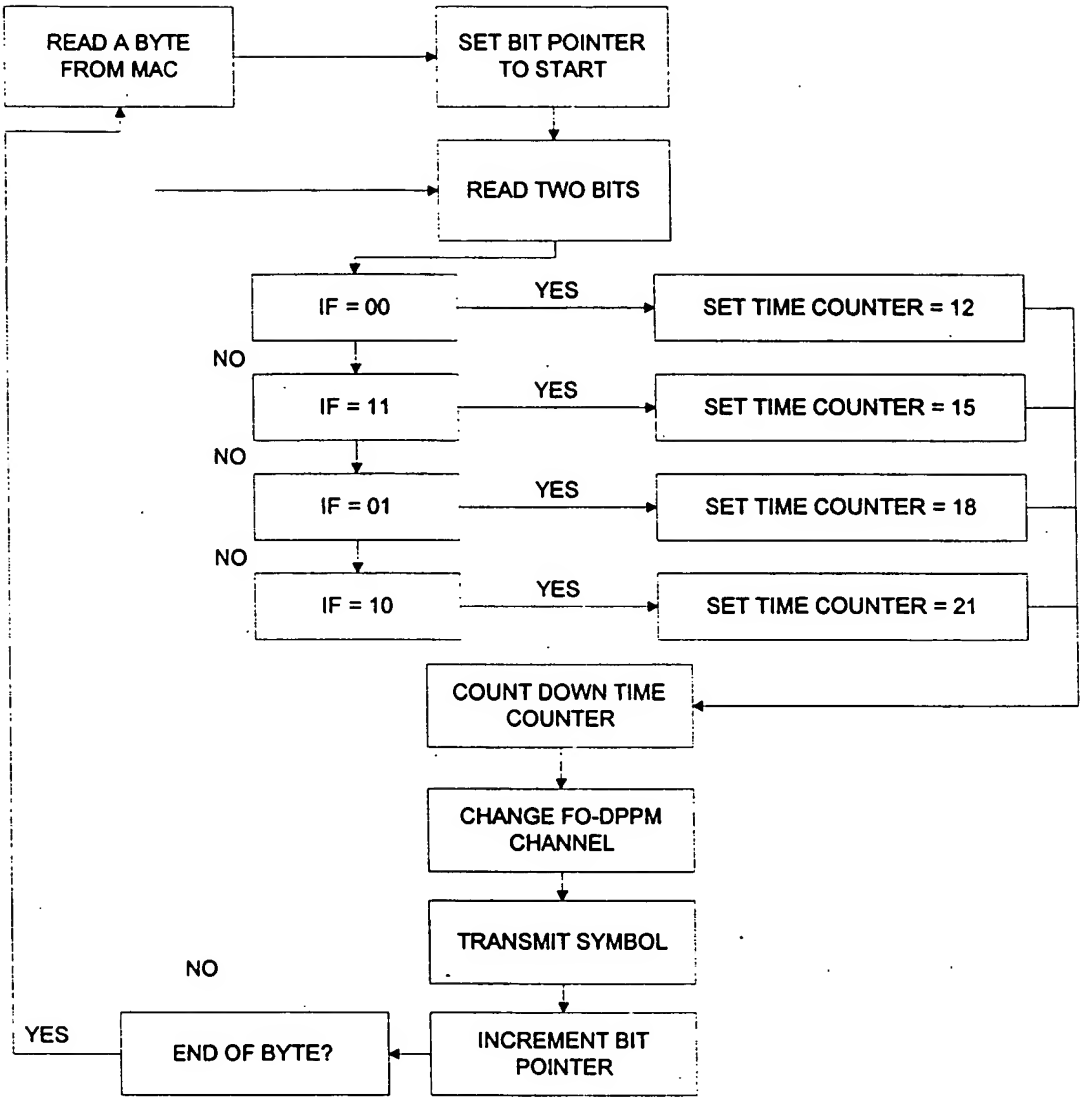
READ A BYTE FROM MAC
SET BIT POINTER TO START
READ TWO BITS
IF = 00
YES
SET TIME COUNTER = 12
NO
IF = 11
YES
SET TIME COUNTER = 15
NO
IF = 01
YES
SET TIME COUNTER = 18
NO
IF = 10
YES
SET TIME COUNTER = 21
COUNT DOWN TIME COUNTER
CHANGE FO-DPPM CHANNEL
TRANSMIT SYMBOL
NO
YES
END OF BYTE?
INCREMENT BIT POINTER

Figure 5B

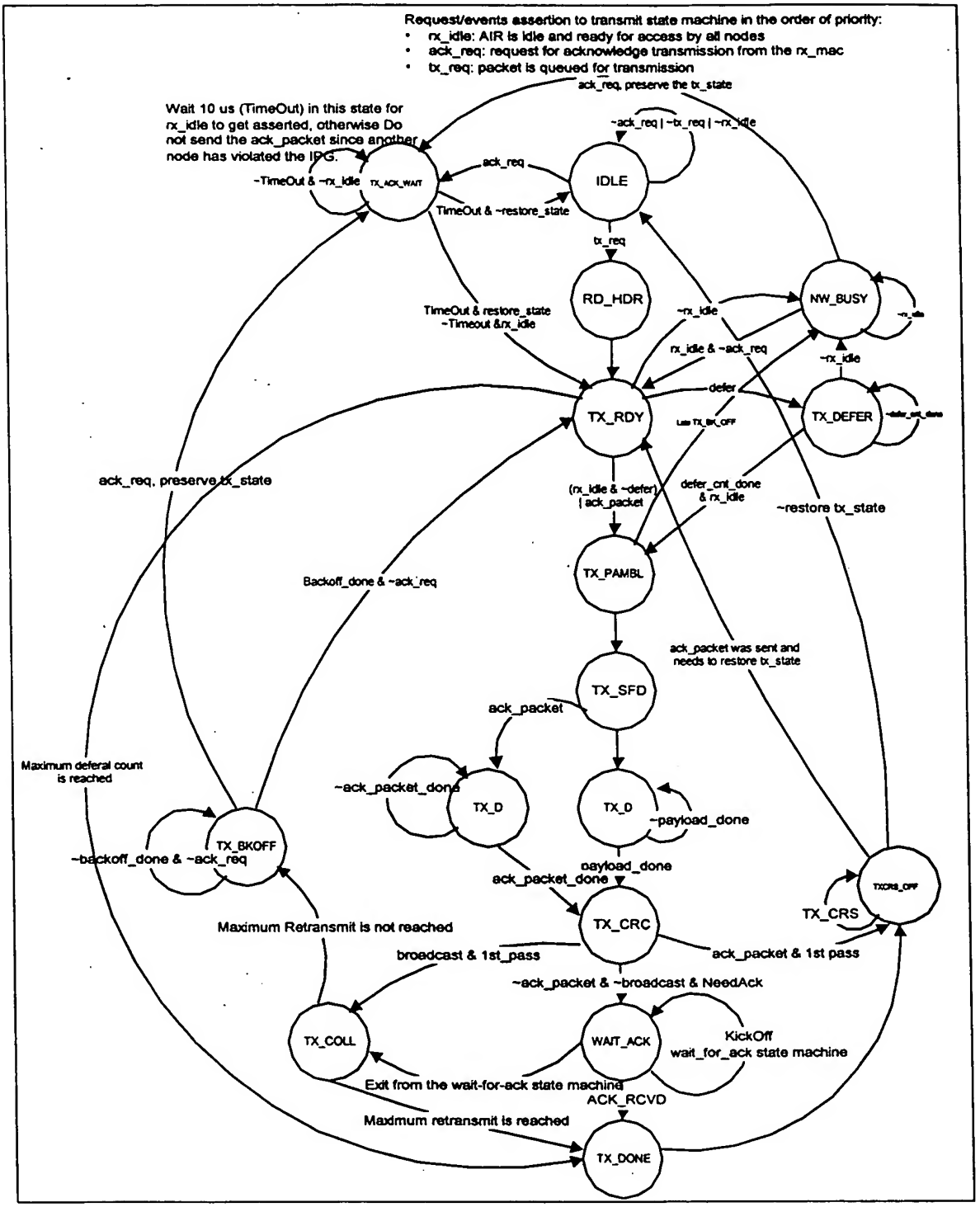
Request/events assertion to transmit state machine in the order of priority:
• rx_idle: AIR is idle and ready for access by all nodes
• ack_req: request for acknowledge transmission from the rx_mac
• tx_req: packet is queued for transmission
ack_req, preserve the tx_state
Wait 10 us (TimeOut) in this state for rx_idle to get asserted, otherwise Do not send the ack_packet since another node has violated the IPG.
~ack_req | ~tx_req | ~rx_idle
ack_req
~TimeOut & ~rx_idle
TX_ACK_WAIT
IDLE
TimeOut & ~restore_state
tx_req
RD_HDR
NW_BUSY
TimeOut & restore_state
~Timeout &rx_idle
~rx_idle
rx_idle & ~ack_req
~rx_idle
defer
TX_RDY
TX_DEFER
ack_req, preserve tx_state
(rx_idle & ~defer) | ack_packet
defer_cnt_done & rx_idle
~restore tx_state
TX_PAMBL
Backoff_done & ~ack_req
ack_packet was sent and needs to restore tx_state
TX_SFD
ack_packet
Maximum deferal count is reached
~ack_packet_done
TX_D
TX_D
~payload_done
payload_done
ack_packet_done
TX_BKOFF
~backoff_done & ~ack_req
TX_CRC
TX_CRS
Maximum Retransmit is not reached
broadcast & 1st_pass
ack_packet & 1st pass
~ack_packet & ~broadcast & NeedAck
TX_COLL
WAIT_ACK
KickOff wait_for_ack state machine
Exit from the wait-for-ack state machine
ACK_RCVD
Maximum retransmit is reached
TX_DONE

Figure 5C

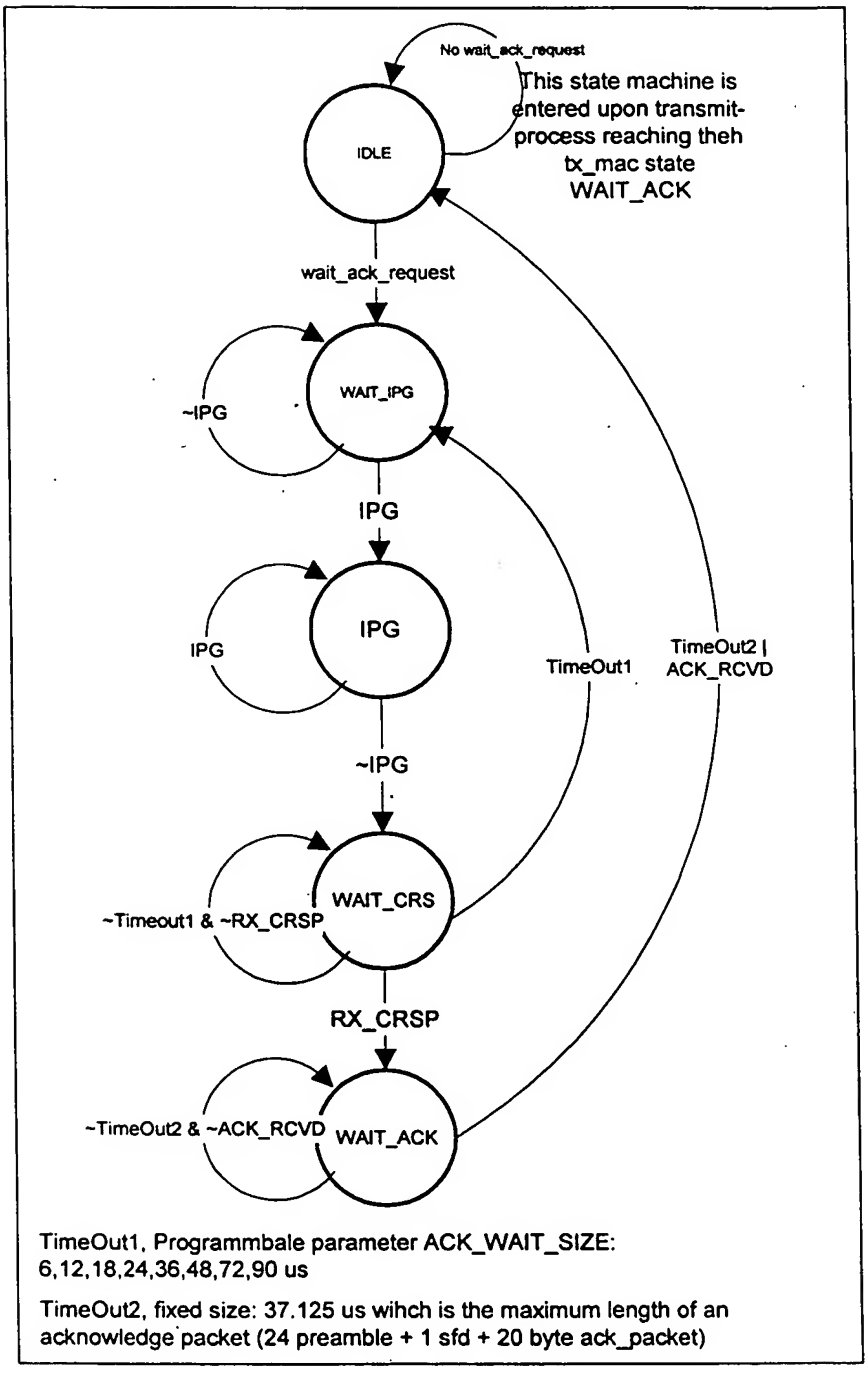
No wait_ack_request
This state machine is entered upon transmit-process reaching theh tx_mac state WAIT_ACK
IDLE
wait_ack_request
WAIT_IPG
~IPG
IPG
IPG
IPG
~IPG
TimeOut1
TimeOut2 | ACK_RCVD
WAIT_CRS
~Timeout1 & ~RX_CRSP
RX_CRSP
~TimeOut2 & ~ACK_RCVD
WAIT_ACK
TimeOut1, Programmbale parameter ACK_WAIT_SIZE: 6,12,18,24,36,48,72,90 us
TimeOut2, fixed size: 37.125 us wihch is the maximum length of an acknowledge packet (24 preamble + 1 sfd + 20 byte ack_packet)

Figure 5D

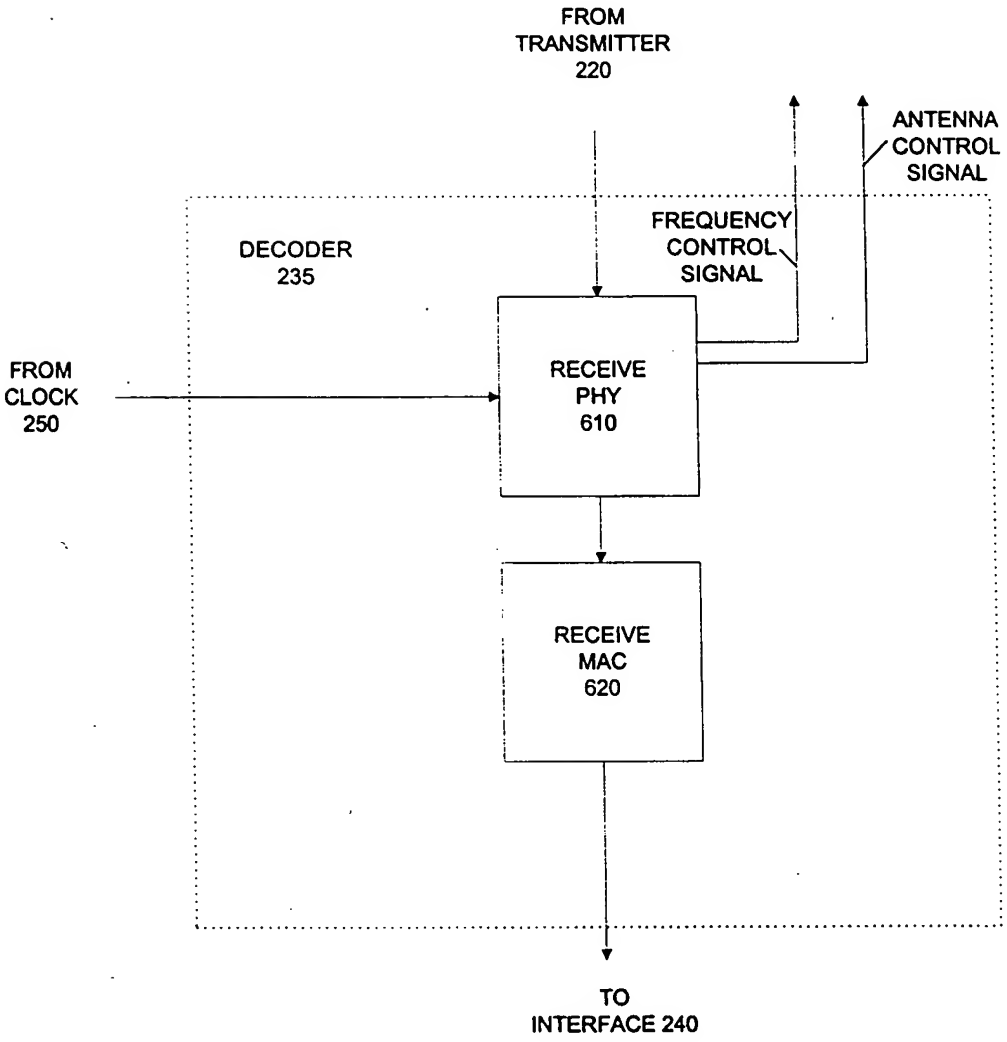
FROM
TRANSMITTER
220
ANTENNA
CONTROL
SIGNAL
FREQUENCY
CONTROL
SIGNAL
DECODER
235
FROM
CLOCK
250
RECEIVE
PHY
610
RECEIVE
MAC
620
TO
INTERFACE 240

Figure 6A

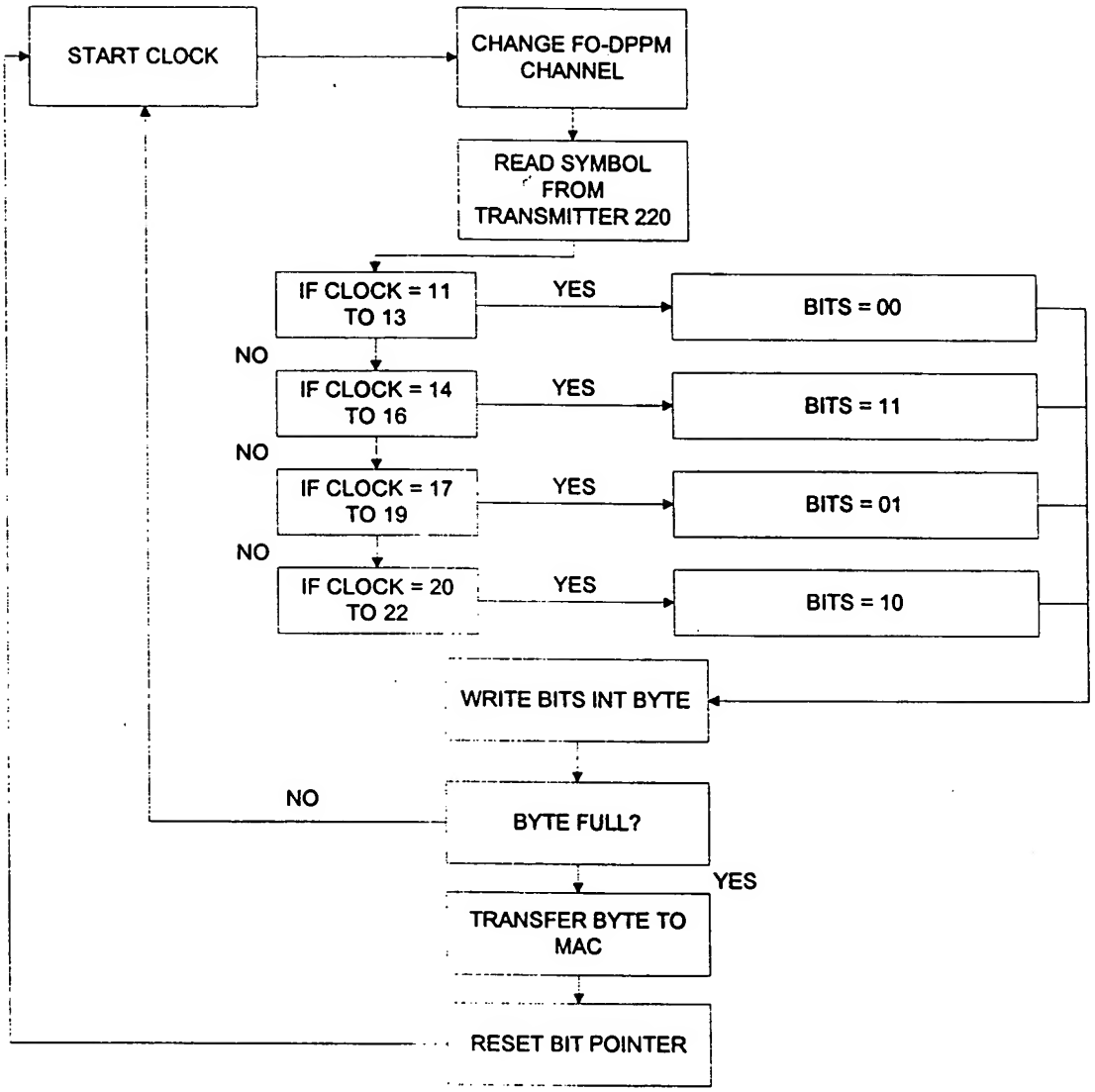
START CLOCK
CHANGE FO-DPPM CHANNEL
READ SYMBOL FROM TRANSMITTER 220
IF CLOCK = 11 TO 13
YES
BITS = 00
NO
IF CLOCK = 14 TO 16
YES
BITS = 11
NO
IF CLOCK = 17 TO 19
YES
BITS = 01
NO
IF CLOCK = 20 TO 22
YES
BITS = 10
WRITE BITS INT BYTE
NO
BYTE FULL?
YES
TRANSFER BYTE TO MAC
RESET BIT POINTER

Figure 6B

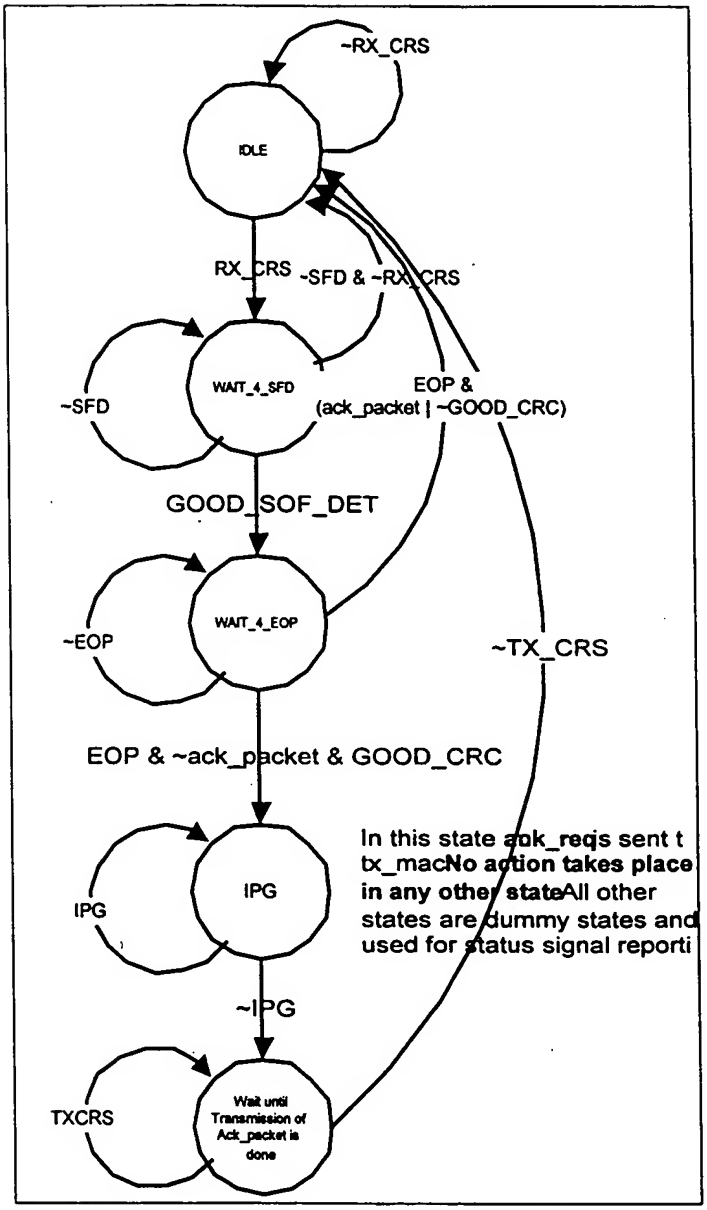
~RX_CRS
IDLE
RX_CRS
~SFD & ~RX_CRS
WAIT_4_SFD
~SFD
EOP &
(ack_packet | ~GOOD_CRC)
GOOD_SOF_DET
WAIT_4_EOP
~EOP
~TX_CRS
EOP & ~ack_packet & GOOD_CRC
IPG
IPG
In this state ack_reqs sent t
tx_macNo action takes place
in any other stateAll other
states are dummy states and
used for status signal reporti
~IPG
TXCRS
Wait until
Transmission of
Ack_packet is
done

Figure 6C

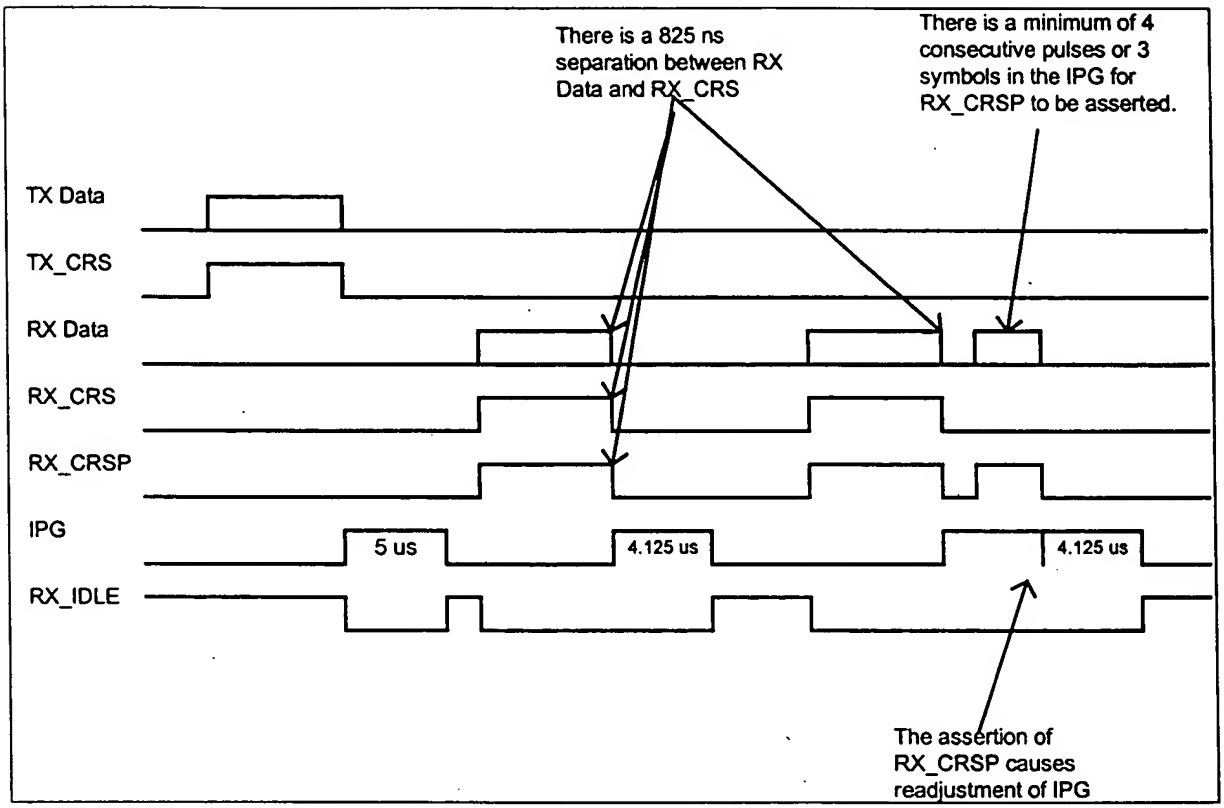
There is a 825 ns separation between RX Data and RX_CRS
There is a minimum of 4 consecutive pulses or 3 symbols in the IPG for RX_CRSP to be asserted.
TX Data
TX_CRS
RX Data
RX_CRS
RX_CRSP
IPG
5 us
4.125 us
4.125 us
RX_IDLE
The assertion of RX_CRSP causes readjustment of IPG

Figure 6D

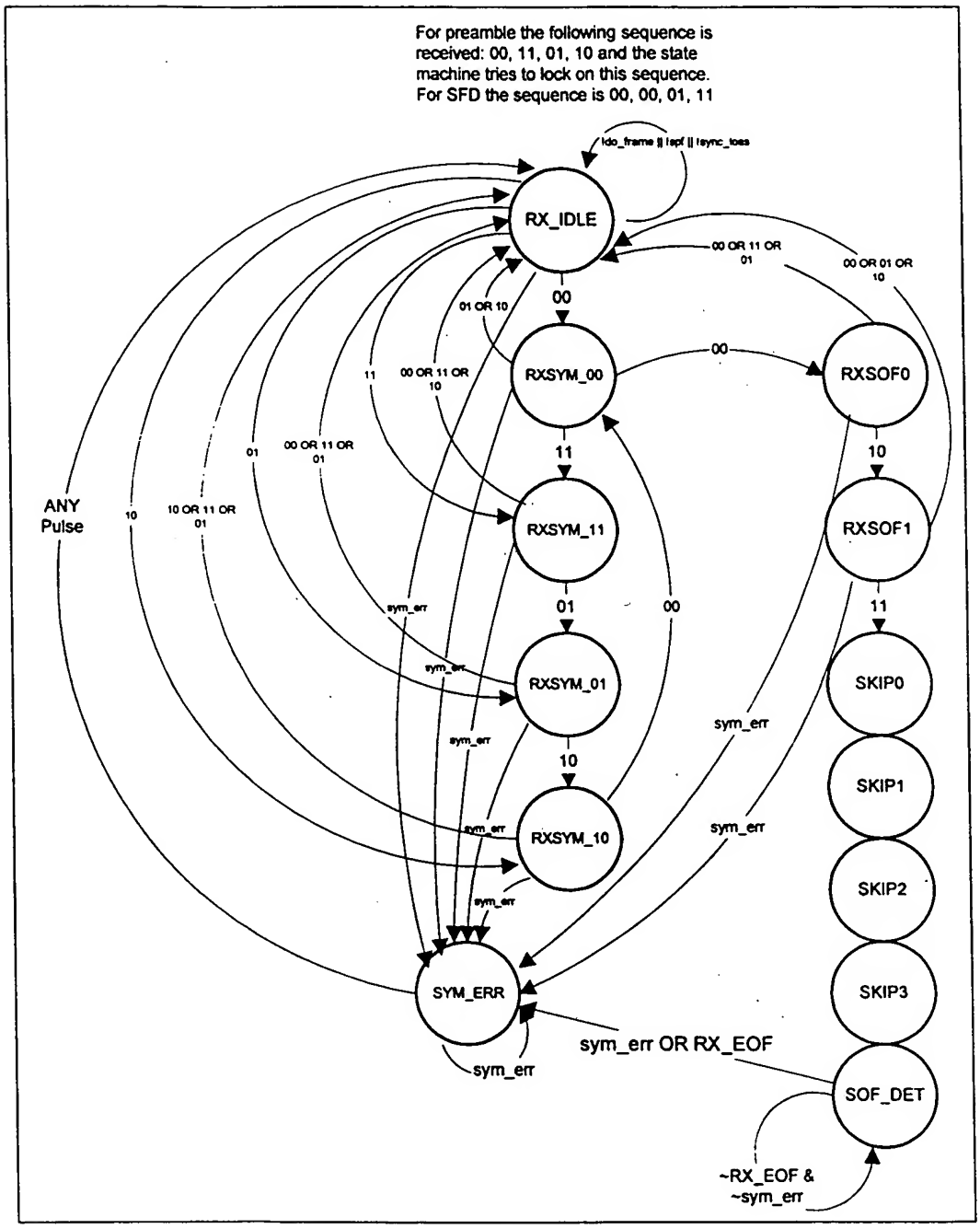
For preamble the following sequence is received: 00, 11, 01, 10 and the state machine tries to lock on this sequence.
For SFD the sequence is 00, 00, 01, 11
!do_frame || !spf || !sync_loss
RX_IDLE
00
01 OR 10
00 OR 11 OR 01
00 OR 01 OR 10
RXSYM_00
00
RXSOF0
11
00 OR 11 OR 10
01
00 OR 11 OR 01
10
ANY Pulse
10 OR 11 OR 01
RXSYM_11
RXSOF1
01
00
11
sym_err
RXSYM_01
SKIP0
10
sym_err
SKIP1
RXSYM_10
SKIP2
SKIP3
SYM_ERR
sym_err OR RX_EOF
sym_err
SOF_DET
~RX_EOF & ~sym_err

Figure 6E